SINGLE LEVEL MIP FILTERING ALGORITHM FOR ANISOTROPIC TEXTURING

CROSS REFERENCE TO RELATED APPLICATION

[0001] This application claims priority to U.S. Provisional Application Serial No. 60/448,956, filed February 21, 2003, and entitled "SINGLE LEVEL MIP FILTERING ALGORITHM FOR ANISOTROPIC TEXTURING," which application is incorporated herein by reference.

FIELD OF THE INVENTION

[0002] The present invention relates to generally to texture mapping and more particularly to anisotropic texture mapping methods designed to reduce aliasing effects.

DESCRIPTION OF THE RELATED ART

[0003] Access to pixel buffers in a graphics system has opened up new ways of processing graphical images. For example, access the pixel buffers in a graphics system permits direct modification of the pixels to achieve certain effects, such as texturing a surface, without the cost of performing these effects in the geometry engines of the graphics system. In the forward direction, texturing a surface typically involves mapping a given point in a 2-dimensional texture space to the surface of a 3-dimensional object in object space. This step is called parameterization. The object is then projected, in a step called a projective transformation, onto a 2-dimensional display in screen space whose resolution is limited by the pixel buffers used to create the display image. Texturing can also be performed in the reverse direction, called inverse mapping. In inverse mapping, for every pixel in screen space, a pre-image of the pixel in the texture space must be found. A square pixel has, in general, a curvilinear quadrilateral pre-image.

[0004] One problem that can occur when performing texture mapping is aliasing. This occurs as a result of the limited resolution of the pixel buffer for the display and the limited size of the texture array that holds the 2-dimensional texture. The limited resolution of the pixel buffer for the display has an inherent spatial sampling frequency and the limited resolution of the texture map has an inherent spatial frequency. If the spatial frequency of the texture map is greater than the spatial sampling frequency inherent in the display pixel buffer, then aliasing can occur.

[0005] There are several conditions under which the spatial frequency of the texture map can be greater than the spatial sampling frequency of the display pixel buffer. One condition is that the

object being textured is a distant object or is viewed in perspective, thereby reducing the size of the object to a small number of pixels compared with a full-size orthographic view of the object in screen space. Another condition is that the size of a window through which the object is viewed in the display space is small. In either case, aliasing occurs, thereby distorting texture and the upsetting the realism sought to be achieved. It is therefore desirable to minimize aliasing under these and other similar circumstances.

[0006] A well-known technique for minimizing aliasing of mapped textures, called mip-mapping, is to provide a series of texture arrays, the highest level texture array in the series having the greatest resolution, and the lowest array in the series having the lowest resolution, with intermediate arrays having intermediate resolutions. Each lower dimension texture map is a filtered version of a next higher dimension of the texture array, so that there are smooth transitions between levels in the series of texture arrays. A key issue in this technique is determining which of the series of texture arrays should be chosen to supply the source of the texture. If a level too high is chosen, then aliasing will result, and if too low a level is chosen an array is chosen, the texture will not show up clearly. The parameter that describes the level chosen is called LOD (Level Of Detail).

[0007] Filtering of the lower level of detail texture arrays involves the concept of a footprint or area of support. This is the area over which an average or some other filtering function is computed on the next higher level of detail. A common type of footprint is a circle, which implies that the filtering function is the same in each dimension of a 2-dimensional texture array. Another more general footprint is an ellipse, which implies anisotropy because the major axis is different from the minor axis.

[0008] The amount of anisotropy can be quantified by a ratio, $\frac{\max\left\|\frac{\partial u}{\partial s},\frac{\partial v}{\partial s}\right\|}{\min\left\|\frac{\partial u}{\partial s},\frac{\partial v}{\partial s}\right\|}$, where (u,v) are coordinates in a 2-dimensional texture space, (x,y) are coordinates in the display space, and s =

$(\cos\alpha, \sin\alpha)$, and $\frac{\partial u}{\partial s} = \frac{\partial u}{\partial x}\cos\alpha + \frac{\partial u}{\partial y}\sin\alpha$, as described in U.S. Patent No. 6,219,064 (Kamen et al.).

[0009] As a standard mathematical approach, the footprint of a texture map on a pixel can be derived from the Jacobian matrix $J = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{pmatrix}$. The footprint is an ellipse where the two axes are in the direction of du/dx, dv/dx and du/dy and dv/dy, respectively. The magnitude of the axes determines the footprint area and shape.

[00010] In isotropic filtering, the differences between axes is ignored and the maximum axis is used to calculate the level of detail (LOD). The LOD is computed as $d = \log_2$(magnitude of long axis). This results in a poor quality image when the ratio of the two axes is large.

[00011] In anisotropic filtering, the differences of the axes is considered. More texels are sampled along the long axis than along the short axis and the LOD is computed as $d = \log_2$(magnitude of short axis). Thus, by applying a proper filter, the resulting texture from multiple texels can faithfully represent the pixel color. If, along the short axis, the footprint covers exactly one texel at a corresponding mip-map level, then only that mip-map level is needed for filtering. However, if the footprint in the short axis direction covers more than just one texel, more samples along the short axis are needed for proper filtering. The current art interpolates between the current mip-map level and next lower resolution mip-map level to obtain the final color of the pixel. However, this requires access to multiple mip-map levels, with an associated performance loss. Access to a second level of the mip-map always costs additional bus transaction cycles which slow down the texture filtering processing. In an efficient cache design, in which one level filtering can be performed in one cycle, additional memory cycles for second level processing reduces the performance of anisotropic filtering by half. The problem manifests itself more when the degree of anisotropy is large. Therefore, it is desirable to generate a high quality textured image when the degree of anisotropy is large, but without incurring a performance loss.

BRIEF SUMMARY OF THE INVENTION

[00012] The present invention uses only one level of the mip-map to perform the filtering without regard to the amount of coverage of the footprint of a pixel on the texture. This is done by deriving the next lower-resolution mip-map level from the current level for anisotropic filtering purposes. Using one level instead of two avoids the loading of the next level texels into the cache, which takes an additional memory cycle. In addition, the cache hit rate is improved, because the current level texels has locality of reference.

[00013] A method in accordance with the present invention is a method of performing anisotropic mip-mapping. The method includes mapping a target pixel needing texture to one or more texels in a higher resolution texture array, where a region of support in the higher resolution texture array is defined by a long and a short axis, is generally elliptical and a level of detail is derived from the short axis. The method further includes performing a filtering function along an axis using the texels from the higher resolution texture array to simulate a filtering effect of using texels from the higher resolution texture array and a second texel array having a lower resolution. The step of performing a filtering function includes, in one embodiment, using the texels from the higher resolution texture array to derive texels of the lower resolution texture array, interpolating the texels from the higher resolution texture array to form a first blended texel, interpolating the texels from the lower resolution texel array to form a second blended texel, and, interpolating the first blended and second blended texels to arrive at a texture for the target pixel.

[00014] One advantage of the present invention is that few texture cache reads are needed for perform a multilevel filtering function.

[00015] Another advantage is that a multilevel filtering function can always be performed in the processing of pixels, because no performance loss occurs.

BRIEF DESCRIPTION OF THE DRAWINGS

[00016] These and other features, aspects and advantages of the present invention will become better understood with regard to the following description, appended claims, and accompanying drawings where:

FIG. 1 shows prior art texture mapping;

FIG. 2 shows texture mapping in accordance with the present invention;

FIG. 3A shows the one sample case of reading a cache line (best case) in the prior art;

FIG. 3B shows the one sample case of reading a cache line (best case) in the present invention;

FIG. 4A shows the one sample case of reading a cache line (worst case) in the prior art;

FIG. 4B shows the one sample case of reading a cache line (worst case) in the present invention;

FIG. 5A shows the two samples case of reading a cache line (best case) in the prior art;

FIG. 5B shows the two samples case of reading a cache line (best case) in the present invention;

FIG. 6A shows the two samples case of reading a cache line (worst case) in the prior art;

FIG. 6B shows the two samples case of reading a cache line (worst case) in the present invention;

FIG. 7A shows the eight samples case of reading a cache line in the prior art;

FIG. 7B shows the eight samples case of reading a cache line in the present invention;

FIG. 8 shows a flow chart of the overall steps in the present invention; and

FIG. 9A shows, graphically, the parameters involved for the one dimension case in accordance with the present invention;

FIG. 9B shows, graphically, the parameters involved for the two dimension case in accordance with the present invention;

FIG. 10 shows a block diagram of a graphics system for practicing an embodiment of the present invention; and

FIG. 11 shows a block diagram of the texture cache and a preferred embodiment of the texture-mapping engine in more detail.

DETAILED DESCRIPTION OF THE INVENTION

[00017] FIG. 1 shows prior art texture mapping as described in U.S. Patent 6,304,268 (Iourcha) and assigned to S3 Graphics Co. LTD., which patent is hereby incorporated by reference in to the present specification. Level n1 has two samples A1 and A2 and level n2 has two samples B1 and B2. A1 and B1 are blended to produce C1 and A2 and B2 are blended to produce C2. Then C1 and C2 are summed with weights to form the final color.

[00018] FIG. 2 shows texture mapping in accordance with the present invention. In FIG. 2 all eight samples in one level are summed with weights to form the final color. Two level mip-maps are not used.

[00019] FIG. 3A shows the one sample case of reading a cache line (best case) in the prior art. In FIG. 3A two cache lines are fetched to read the two levels of the mip-map.

[00020] FIG. 3B shows the one sample case of reading a cache line (best case) in the present invention. In FIG. 3B, one cache line is fetched to read the single level needed of the mip-map.

[00021] FIG. 4A shows the one sample case of reading a cache line (worst case) in the prior art. In FIG. 4A, three cache lines are fetched to obtain the two levels of mip-map needed.

[00022] FIG. 4B shows the one sample case of reading a cache line (worst case) in the present invention. In FIG. 4B, two cache lines are fetched for the single level needed.

[00023] FIG. 5A shows the two samples case of reading a cache line (best case) in the prior art. In FIG. 5A, two cache lines are fetched, one for each level.

[00024] FIG. 5B shows the two samples case of reading a cache line (best case) in the present invention. In FIG. 5B, one cache line is fetched for the single needed level.

[00025] FIG. 6A shows the two samples case of reading a cache line (worst case) in the prior art. In FIG. 6A, three cache lines are fetched to obtain the levels needed.

[00026] FIG. 6B shows the two samples case of reading a cache line (worst case) in the present invention. In FIG. 6B, two cache lines are fetched for the single level needed.

[00027] FIG. 7A shows the eight samples case of reading a cache line in the prior art. In FIG. 7A, five cache lines are read to obtain eight samples.

[00028] FIG. 7B shows the eight samples case of reading a cache line in the present invention. In FIG. 7B, three cache lines are read to obtain the eight samples.

[00029] FIG. 8 shows a flow chart of the overall steps in the present invention. If two-level mip-mapping is used, then cache addressing scheme 1 is used and one higher resolution mip-map level is used to simulate two levels of mip-mapping. If two-levels mip-mapping is not used, then addressing scheme 2 is used.

In accordance with the present invention, the formula for filtering 4 texels, in one dimension, is

$$\text{final color} = Wa*ColorA + Wb*ColorB + Wc*ColorC + Wd*ColorD,$$

where Wa, Wb, Wc and Wd are the filter weights for each of the colors ColorA, ColorB, ColorC, ColorD of the four texels. The filter weights are:

$$Wa = 1/8\ (1-Df)(1+2Uf),$$

$$Wb = Uf*Df + Wa,$$

$$Wc = (1-Uf)*Df + Wd, \text{ and}$$

$$Wd = 1/8\ (1-Df)(3-2Uf)$$

where Uf is the fraction of the U texture coordinate and Df is the fraction of the LOD.

For the V coordinate, Uf is replaced with Vf.

[00030] FIG. 9A shows, graphically, the parameters involved for the one dimension case in accordance with the present invention. Level n+1 is a higher resolution texel level and level n is a lower resolution texel level. The distance between the levels is unity. An arbitrary pixel P is located at some level between level n+1 and level n. Pixel P's position from the lower resolution level is controlled by parameter Df. Pixel P's position from the higher resolution level is 1-Df. Pixel P maps to a position P_{n+1} in level n+1 and position P_n in level n.

[00031] Parameter Uf gives the distance from the center of texel C (T_C) to the P_{n+1} position and 1-Uf gives the distance from the center of texel B (T_B), where Uf is the fraction of the U texture coordinate and where unity is assumed to be the distance between texel centers. Parameter h gives the distance from the edge of T_B to the position P_{n+1}.

[00032] Parameter m gives the distance from the center of texel CD (T_{CD}) to the P_n position and n = (1-m) = (1-(k+1/2)) gives the distance from the center of texel AB (T_{AB}). Parameter k gives the distance from the edge of T_{CD} to the position P_n.

[00033] Based on the above, the color at point P_{n+1} is a linear interpolation of the colors at TB and TC, based on position parameter Uf.

$$TP_{n+1} = U_f T_B + (1-U_f)T_C.$$

The color at P_n is a linear interpolation of the colors at T_{AB} and T_{CD} based on position parameter m,

$$TP_n = mT_{AB} + (1-m)T_{CD}.$$

The color TP at P is then a linear interpolation of TPn and TPn+1 based on position parameter q between the levels,

$$TP = (1-q)TP_{n+1} + qTP_n$$

Substituting the TPn and TPn+1 expressions into TP gives

$$TP = (1-q)U_f T_B + (1-q)(1-U_f)T_C + qmT_{AB} + q(1-m)T_{CD}$$

Because only colors TA, TB, TC and TD are available from the cache memory, which stores level n+1, the colors from level n are derived from these colors as follows:

$$T_{AB} = (T_A + T_B)/2$$

$$T_{CD} = (T_C + T_D)/2$$

Therefore,

$$TP=(1-q)U_f T_B + (1-q)(1-U_f)T_C + qm(T_A+T_B)/2 + q(1-m)(T_C+T_D)/2.$$

Collecting like terms yields,

$$TP=T_A(qm/2) + T_B((1-q)U_f + qm/2) + T_C((1-q)(1-U_f) + q(1-m)/2) + T_D(q(1-m)/2).$$

Therefore, the weights are

$$W_a = 1/8\cdot(1-D_f)(1+2U_f)$$

$$Wb = D_f U_f + W_a,$$

$$W_c = D_f(1-U_f) + W_d, \text{ and}$$

$$W_d = 1/8\cdot(1-D_f)(3-2U_f),$$

where the equalities k = h/2, m = (h+1)/2, h = Uf-1/2, and n = 1-m, were used.

Note that $\sum_i W_i = D_f U_f + 2W_a + D_f(1-U_f) + 2W_d = D_f + 2(W_a + W_d)$. Therefore,

$$\sum_i W_i = D_f + \frac{1}{4}(1 - Df)(4) = 1$$

[00034] FIG. 9B shows, graphically, the parameters involved for the two dimension case in accordance with the present invention. Geometry similar to the U dimension is introduced for the V dimension, where Vf gives the position of the texel in the higher resolution texel map. In the two dimensional case, V direction also has four samples, so both U and V dimensions form a 4x4 texel 2D array Wuv. Each one of the 4x4 texels has a corresponding weighting factor taken from the Wuv array. The weighted sum is then the final color for the anisotropic sample. In practice, one can tabulate the weight array Wuv according to the Df, Uf, Vf and a desired filtering function. Below is one of them, when box filtering is assumed (similar to the discussions above for one dimension case):

Wuv= Duv*Uuv*Vuv, u=0,1,2,3, v=0,1,2,3

Duv= 0.25* Df, if either u=0 or 3, or v=0, or 3, (1-0.75)*Df, otherwise

Uuv= (1-Uf), if u=0 or u=1,

Uuv= Uf, if u=2 or u=3,

Vuv= 1-Vf, if v=0 or v=1, and

Vuv =Vf , if v=2 or v=3.

The complete Wuv array is set forth below.

$$W_{uv} = \begin{bmatrix} 0.25D_f(1-U_f)(1-V_f) & (1-0.75D_f)(1-U_f)(1-V_f) & (1-0.75D_f)U_f(1-V_f) & 0.25D_fU_f(1-V_f) \\ (1-0.75D_f)(1-U_f)(1-V_f) & (1-0.75D_f)(1-U_f)(1-V_f) & (1-0.75D_f)U_f(1-V_f) & (1-0.75D_f)U_f(1-V_f) \\ (1-0.75D_f)(1-U_f)V_f & (1-0.75D_f)(1-U_f)V_f & (1-0.75D_f)U_fV_f & (1-0.75D_f)U_fV_f \\ 0.25D_f(1-U_f)V_f & (1-0.75D_f)(1-U_f)V_f & (1-0.75D_f)U_fV_f & 0.25D_fU_fV_f \end{bmatrix}$$

The final color = Σ(Wuv*Color(u,v)), where u and v each take on values from 0 to 3, and where Color(u,v) is the color (texel) corresponding to Wuv on the read mipmap level. Also please note that $\sum W_{uv} = 1$, where u=0, 1, 2, 3 and v = 0, 1, 2, 3.

[00035] In the case of multiple anisotropic samples, the weights of each sample also depend on the anisotropic ratio, and the filter function chosen for the anisotropic case. For example, if the ratio is 3.5, then 4 samples can be chosen. Assuming a box filtering function, then the weight for each anisotropic sample is 0.25. If a Gaussian filtering function is assumed, then the weight for each sample is subject to the function exp(-((x-x0)(x-x0)/ratio*ratio), where x0 is

the cluster center of N samples, and x is the location of the sample. The sample weights must be normalized to 1. The sampling point of samples is across the region of anisotropic shape at the long axis.

[00036] In general, the equation for anisotropic filtering can be expressed as follows.

$$\text{final color} = \sum WS_i * ColorS_i = 1$$

where i takes on values from 1 to the number of samples, ΣWSi=1 for i from 1 to the number of samples, WSi is a weighting function for the ith sample, and ColorSi is the blended color for the ith sample. The greater the number of samples chosen, the better the quality of the image is, but there is a performance loss. Since only the high resolution mip-map level is sampled, there is a tradeoff between quality and performance.

[00037] Generally speaking, in the one dimension case, a single texel in the lower resolution mipmap level is derived from two adjacent texels in the higher resolution mipmap level, and two adjacent texels in the lower resolution mipmap level are derived from four adjacent texels in the higher resolution mipmap level. Because a pixel at an LOD between these two mipmap levels is blended from two adjacent texels in the lower resolution mipmap level and two adjacent texels in the higher resolution mipmap level, such a pixel can therefore be blended from four adjacent texels in the higher resolution mipmap level in accordance with the present invention. Please refer to FIG. 9 and the above equations, in which four adjacent texels of the same mipmap level are used.

[00038] In the two dimension case, one texel in the lower resolution mipmap level is derived from four texels in a 2 x 2 block in the higher resolution mipmap level, and four texels in a 2 x 2 block in the lower resolution mipmap level are derived from sixteen adjacent texels in a 4 x 4 block in the higher resolution mipmap level. Because a pixel at an LOD between these two mipmap levels is blended from four texels in the 2 x 2 block in the lower resolution mipmap level and four texels in the 2 x 2 block in the higher resolution mipmap level, such a pixel can therefore be blended from sixteen texels in a 4 x 4 block in the higher resolution mipmap level in accordance with the present invention. This can be understood by referring to the above weight array Wuv, which indicates how sixteen texels of the same mipmap level are used.

[00039] FIG. 10 shows a block diagram of a graphics system for practicing an embodiment of the present invention. System 800 includes a texture memory 802, a texture cache 804, a texture mapping engine 806, a pixel processing module 808, a primitives memory 810, a frame buffer memory 812, and a display device 814. The texture mapping engine, texture cache and pixel processing module are often included on a graphics acceleration chip. The texture memory 802 stores two-dimensional representations of texture to be mapped onto primitives. In the present invention, preferably one level of texture detail is stored. The texture cache 804 provides temporary storage of portions of the texture memory 802. The graphic primitives are preferably stored in a primitives memory 810, which contains information describing the size and shape of graphics elements such as triangles or other polygons to be displayed. The texture mapping engine performs the operation of mapping textures stored in texture memory 802 onto primitives stored in the primitives memory 810. The output of the texture mapping engine 806 is connected to an input of the pixel processing module 808. The pixel processing module performs z-buffering, texture lighting, fogging, alpha-blending, and other pixel operations and writes the resulting rendered image to the frame buffer memory 812. The image in the frame buffer is sent to the display device such as a CRT or LCD panel.

[00040] FIG. 11 shows a block diagram of the texture cache and a preferred embodiment of the texture-mapping engine in more detail. The texture mapping engine 806 includes a first level generator 1202, an interpolator 1204, and a sample producer 1206 and a second level generator 1208. The first level generator is coupled to the texture cache 804, and generate a first level of detail n+1. The output of the first level generator 1202 is coupled to a first input of the interpolator 1204 to provide a blended higher resolution texel. The sample producer 1206 is coupled to the texture cache to receive stored texels and produces texels for the next level n of detail. The output of the sample producer 1206 is coupled to in input of the second level generator 1208. The second level generator 1208 uses the lower resolution texels to produce blended lower resolution texel. The interpolator 1204 generates the final texture value from the blended higher resolution texel and the blended lower resolution texel based on the level of detail (LOD) parameter.

[00041] Although the present invention has been described in considerable detail with reference to certain preferred versions thereof, other versions are possible. Therefore, the spirit

and scope of the appended claims should not be limited to the description of the preferred versions contained herein.